Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Theoretical Computer Science 304 (2003) 157–183

Theoretical
Computer Science

www.elsevier.com/locate/tcs

The complexity of bisimilarity-checking for one-counter processes

Antonín Kučera[1]

Faculty of Informatics, Masaryk University, Botanická 68a, 60200 Brno, Czech Republic

Received 11 June 2001; received in revised form 14 January 2003; accepted 17 January 2003
Communicated by M. Jerrum

Abstract

We study the problem of bisimilarity-checking between processes of one-counter automata and finite-state processes. We show that deciding weak bisimilarity between processes of one-counter nets (which are 'restricted' one-counter automata where the counter cannot be tested for zero) and finite-state processes is **DP**-hard. In particular, this means that the problem is both **NP** and **co-NP** hard. The same technique is used to demonstrate **co-NP**-hardness of strong bisimilarity between processes of one-counter nets. Then we design an algorithm which decides weak bisimilarity between processes of one-counter automata and finite-state processes in time which is polynomial for a large subclass of instances, giving a kind of characterization of all hard instances as a byproduct. Moreover, we show how to efficiently estimate the time which is needed to solve a given instance. Finally, we prove that the problem of strong bisimilarity between processes of one-counter automata and finite-state processes is in **P**.

Keywords: One-counter automata; One-counter nets; Bisimilarity

1. Introduction

In concurrency theory, *processes* are typically understood as (being associated with) states in *transition systems*, a fundamental and widely accepted model of discrete systems.

[1] Supported by the Grant Agency of the Czech Republic, grant No. 201/03/1161.
E-mail address: tony@fi.muni.cz (A. Kučera).

0304-3975/03/$ - see front matter
doi:10.1016/S0304-3975(03)00081-1

Definition 1. *A* transition system *is a triple* $\mathscr{T} = (S, \Sigma, \rightarrow)$ *where* S *is a set of* states, Σ *is a finite set of* actions (*or* labels), *and* $\rightarrow \subseteq S \times \Sigma \times S$ *is a* transition relation.

We write $s \xrightarrow{a} t$ instead of $(s,a,t) \in \rightarrow$ and we extend this notation to elements of Σ^* in the natural way. A state t is *reachable* from a state s, written $s \rightarrow^* t$, iff there is $w \in \Sigma^*$ such that $s \xrightarrow{w} t$. A system $\mathscr{T}$ is *finite-state* iff the set of states of $\mathscr{T}$ is finite.

1.1. Formal verification of concurrent systems

The *equivalence approach* to formal verification of concurrent systems is based on the following scheme: One describes the *specification* (the intended behaviour) $\mathscr{S}$ and the *implementation* $\mathscr{I}$ of a given system in some 'higher' formalism whose semantics is defined in terms of transition systems, and then it is shown that $\mathscr{S}$ and $\mathscr{I}$ are *equivalent*. Here, the notion of process equivalence can be captured in many ways (see, e.g., [47]). It seems, however, that *bisimulation equivalence* [40,37] is of special importance because its accompanying theory has been developed very intensively and found its way to many practical applications.

Definition 2. Let $\mathscr{T} = (S, \Sigma, \rightarrow)$ be a transition system. A binary relation $R \subseteq S \times S$ is a bisimulation iff whenever $(s,t) \in R$, then
- for each $s \xrightarrow{a} s'$ there is some $t \xrightarrow{a} t'$ such that $(s',t') \in R$,
- for each $t \xrightarrow{a} t'$ there is some $s \xrightarrow{a} s'$ such that $(s',t') \in R$.

States s,t are *bisimulation equivalent* (or *bisimilar*), written $s \sim t$, iff there is a bisimulation relating them.

Bisimulations can also be used to relate states of *different* transition systems; formally, two systems can be considered as a single one by taking their disjoint union (the labeling of transitions is preserved). An important variant of bisimilarity is *weak bisimilarity* introduced by Milner in his work on CCS [37]. This relation distinguishes between 'external' and 'internal' computational steps, and allows to 'ignore' the internal steps (which are usually denoted by a distinguished action τ) to a certain extent.

Definition 3. Let $\mathscr{T} = (S, \Sigma, \rightarrow)$ be a transition system. We define the *extended transition relation* $\Rightarrow \subseteq S \times \Sigma \times S$ as follows:
- $s \stackrel{\tau}{\Rightarrow} t$ iff t is reachable from s via a finite (and possibly empty) sequence of transitions labeled by τ (note that $s \stackrel{\tau}{\Rightarrow} s$ for each s),
- $s \stackrel{a}{\Rightarrow} t$ where $a \neq \tau$ iff there are states u,v such that $s \stackrel{\tau}{\Rightarrow} u \xrightarrow{a} v \stackrel{\tau}{\Rightarrow} t$.

The relation of *weak bisimulation* is defined in the same way as bisimulation, but '$\Rightarrow$' is used instead of '$\rightarrow$'. Processes s,t are *weakly bisimilar*, written $s \approx t$, iff there is a weak bisimulation relating them.

To prevent a confusion about bisimilarity and weak bisimilarity, we refer to bisimilarity as *strong bisimilarity* in the rest of this paper.

1.2. One-counter automata and one-counter nets

In this paper we study the complexity of certain bisimulation problems for processes of transition systems generated by *one-counter automata* and *one-counter nets*. These models are formally seen as restricted classes of *pushdown automata*.

Definition 4. *A* pushdown automaton *is a tuple* $\mathscr{P}=(Q,\Gamma,\Sigma,\delta)$ *where* Q *is a finite set of* control states, Γ *is a finite* stack alphabet, Σ *is a finite* input alphabet, *and* $\delta : (Q\times\Gamma) \to 2^{\Sigma\times(Q\times\Gamma^*)}$ *is a* transition function *such that* $\delta(p,X)$ *is finite for all* $p\in Q$ and $X\in\Gamma$.

We can assume (w.l.o.g.) that each transition increases the height (or length) of the stack at most by one (every PDA can be efficiently transformed to this kind of normal form). To $\mathscr{P}$ we associate the transition system $\mathscr{T}_{\mathscr{P}}$ where $Q\times\Sigma^*$ is the set of states, Σ is the set of actions, and the transition relation is determined by

$$(p,A\alpha)\xrightarrow{a}(q,\beta\alpha) \quad \text{iff } (a,(q,\beta))\in\delta(p,A).$$

As usual, we write $p\gamma$ instead of (p,γ) and we use ε to denote the empty word. The size of $\mathscr{P}$ is the length of the string which is obtained by writing all elements of the tuple linearly in binary. The size of a process $p\alpha$ of $\mathscr{P}$ is the length of its corresponding binary encoding. Pushdown processes (i.e., processes of pushdown automata) have their origin in theory of formal languages [21]. In the last decade, they attracted further attention as a natural model of sequential systems which is suitable for purposes of formal verification [14–16].

Note that Definition 4 actually introduces the so-called *real-time* PDA, i.e., PDA without ε-transitions. However, it is still possible to model the silent (externally unobservable) computational steps by τ-labeled transitions; the way how weak bisimilarity treats τ-transitions is very similar to the original treatment of ε-transitions (cf., e.g., [21]).

In this paper we mainly concentrate on a subclass of pushdown automata where the stack behaves like a *counter*. The resulting model naturally corresponds to finite-state programs operating on a single unbounded variable. For example, network protocols can maintain the count on how many unacknowledged messages have been sent, a printer spool should know how many processes are waiting in the input queue, etc.

Definition 5. A *one-counter automaton* (*OC automaton*) $\mathscr{A}$ is a pushdown automaton with just two stack symbols I and Z; the transition function δ of $\mathscr{A}$ is a union of functions δ_Z and δ_I where δ_Z: $(Q\times\{Z\}) \to 2^{\Sigma\times(Q\times(\{I\}^*\{Z\}))}$ and δ_I: $(Q\times\{I\}) \to 2^{\Sigma\times(Q\times\{I\}^*)}$.

Hence, Z works like a bottom-of-stack symbol (which cannot be removed), and the number of pushed I's represents the counter value. Processes of $\mathscr{A}$ (i.e., states of $\mathscr{T}_{\mathscr{A}}$) are of the form pI^iZ. In the rest of this paper, we often write $p(i)$ instead of pI^iZ. It is worth to note that the size of $p(i)$ is $\mathcal{O}(i)$ and *not* $\mathcal{O}(\log i)$, because $p(i)$ is just a *symbolic* abbreviation for $p\alpha Z$ where α is a string of i symbols I (i.e., we keep

using the standard measure introduced for PDA). Again, we assume (w.l.o.g) that each transition increases the counter at most by one.

A proper subclass of one-counter automata of its own interest are *one-counter nets*. Intuitively, OC nets are 'restricted' OC automata which cannot test for zero explicitly. They are equivalent to a subclass of Petri nets [41] with (at most) one unbounded place. Here, the word 'equivalent' means that the two models generate the same class of transition systems up to isomorphism.

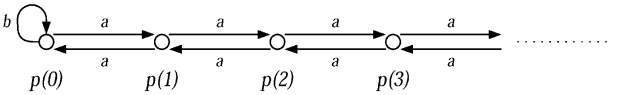
Definition 6. A *one-counter net* $\mathcal{N}$ is a one-counter automaton such that whenever $(a, qI^iZ) \in \delta_Z(p, Z)$, then $(a, qI^{i+1}) \in \delta_I(p, I)$.

In other words, each transition which is enabled at zero-level is also enabled at (each) non-zero-level. Hence, there are no 'zero-specific' transitions which could be used to 'test for zero'.

As a simple example, we might take $\mathcal{A} = (\{p\}, \{I, Z\}, \{a\}, \delta)$, where

$$\delta(pZ) = \{(b, pZ), (a, pIZ)\}, \quad \delta(pI) = \{(a, pII), (a, p\varepsilon)\}.$$

Note that $\mathcal{A}$ is *not* a one-counter net; however, $\mathcal{A}$ becomes an OC net as soon as we delete the (only) b-transition, or, alternatively, add the (b, pI) transition to $\delta(pI)$. The associated infinite-state transition system $\mathcal{T}_{\mathcal{A}}$ looks as follows:



Observe that the out-going transitions of a OC process $q(i)$ where $i > 0$ do not depend on the actual value of i. Hence, the structure of transition systems which are associated with OC automata (and, in particular, with OC nets) is rather regular—they consist of a 'zero pattern' and a 'non-zero pattern' which is repeated infinitely often. Despite this regularity, bisimilarity-checking for OC automata (and even for OC nets) is computationally hard, as we shall see in Section 2.

1.3. The state of the art

In this section we give a short summary of known results about relevant verification problems for PDA and OC automata.

In the context of concurrency theory, the subclass of stateless pushdown automata is usually referred to as 'BPA' (which stands for **B**asic **P**rocess **A**lgebra). Important subclasses of PDA and BPA can be further obtained by an extra restriction of *normedness*. We say that a pushdown automaton $\mathcal{P}$ is *normed* if each configuration $p\alpha$ of $\mathcal{P}$ can empty its stack in a finite number of transitions (i.e., $p\alpha \to^* q\varepsilon$ for some control state q). It is worth to note that the classes of BPA processes and processes of OC

nets are *incomparable* w.r.t. bisimulation semantics. In particular,

- the 'standard' example of a PDA process which is not bisimilar to any BPA process [9], i.e., the process pIZ of a PDA $\mathscr{P}=(\{p,q,r,s\},\{I,Z\},\{a,b,c,d\},\delta)$ where

$$\delta=\{\ pI\xrightarrow{a}pII,\ \ pI\xrightarrow{b}r\varepsilon,\ \ pI\xrightarrow{c}q\varepsilon,\ \ qI\xrightarrow{d}sI,\ \ sI\xrightarrow{d}q\varepsilon,\ \ rI\xrightarrow{d}r\varepsilon\}$$

  is in fact an OC net process;
- the (normed) BPA process X of a stateless PDA $\mathscr{P}=(\{-\},\{X,Y\},\{a,b,c\},\delta)$ where

$$\delta=\{\ X\xrightarrow{a}YX,\ \ X\xrightarrow{a}XX,\ \ X\xrightarrow{b}\varepsilon,\ \ Y\xrightarrow{a}XY,\ \ Y\xrightarrow{a}YY,\ \ Y\xrightarrow{c}\varepsilon\}$$

  is not bisimilar to any OC process. To see this, realize that X can reach 2^n pairwise non-bisimilar states in n transitions, while any OC process $p(i)$ can reach at most $k\cdot(2n+1)$ states in n transitions, where k is the number of control states.

1.3.1. Bisimilarity-checking

The first positive result indicating that the decidability/complexity issues for bisimilarity are substantially different from the ones for language equivalence was obtained by Baeten et al. [5]. They proved that strong bisimilarity is decidable for normed BPA processes. Simpler proofs were given later in [11,22,18], and there is even a polynomial-time algorithm due to Hirshfeld et al. [20]. The decidability result was extended to all BPA processes in [12]. However, the best known algorithm for this generalized case is of elementary complexity [10]; recently, the **PSPACE** lower bound was presented in [43]. The decidability of strong bisimilarity for normed PDA processes was demonstrated by Stirling in [46]. As his result is obtained by a combination of two semi-decision procedures, it does not allow for a reasonable complexity analysis. Another (incomparable) positive result from [23] says that strong bisimilarity is decidable for processes of OC automata. Later, Sénizergues [42] presented a rather involved proof demonstrating that strong bisimilarity is decidable for all PDA processes. The **EXPTIME** lower complexity bound for the problem of strong bisimilarity over (normed) PDA processes was recently given in [31]. The problem of *weak* bisimilarity for PDA is already undecidable [44].

The situation becomes substantially simpler if (at least) one of the two processes being compared is finite-state. In [25] it was shown that the problem of strong/weak bisimilarity (and, in fact, many bisimulation-like equivalences) between processes of some class $\mathscr{C}$ and finite-state processes can be reduced to the model-checking problem with processes of $\mathscr{C}$ and a slightly generalized version of the logic EF in which the '$\diamond$' modality can impose certain constraints on action sequences. This reduction is not polynomial and therefore it cannot be used to transfer complexity bounds; however, it allows to conclude that strong/weak bisimilarity is decidable for a large class of processes known as 'PAD' [36] which properly subsumes all PDA and also PA [6] processes. The problem of strong/weak bisimilarity between PDA and finite-state processes is **PSPACE**-hard [35], and in fact **PSPACE**-complete [31]. Another related result from [33] is that strong/weak bisimilarity between BPA and finite-state processes is in **P**.

1.3.2. Other results for one-counter processes

The problem of *simulation equivalence* [47] between processes of OC nets has been shown decidable in [1]. It was the first (and rather tight) result demonstrating that also simulation equivalence can be decidable in a non-trivial class of infinite-state processes (otherwise, simulation tends to be undecidable [19], unless one of the two processes is a finite-state [32]). A simpler proof was later given in [29] where it is also proved that simulation equivalence between processes of OC automata is already undecidable. The relationship between simulation and bisimulation equivalence is studied in [26] where it is shown that certain 'simulation-problems' for OC nets are effectively reducible to their 'bisimulation-counterparts' for OC automata. Further evidence supporting the claim that OC automata are generally harder to analyze than OC nets is provided in [30]—simulation equivalence with a finite-state process is **co-NP**-hard for processes of OC automata, while the same problem is in **P** for processes of OC nets.

1.4. Plan of the paper

In this paper, we concentrate on the complexity of checking strong and weak bisimilarity between processes of OC automata and FS processes. Our motivation is that the specification or the implementation of a system which is to be verified (see above) can often be specified as a finite-state process. Moreover, a number of 'classical' verification problems (e.g., liveness, safety) can be easily reduced to the problem of weak bisimilarity with a finite-state system. For example, if we want to check that the action a is *live* for a process g (i.e., each state which is reachable from g can reach a state which can emit a), we can rename all actions of g except a to τ and then check weak bisimilarity between g and f where f is a one-state process with the only transition $f \xrightarrow{a} f$.

In Section 2, it is shown that the problem of weak bisimilarity between processes of OC nets and FS processes is **DP**-hard, even for a fixed finite-state process (intuitively, the class **DP** [39] is expected to be somewhat larger than the union of **NP** and **co-NP**; however, it is still contained in the $\Delta_2 = \mathbf{P}^{\mathbf{NP}}$ level of the polynomial hierarchy). Here we have to devise a special technique for encoding, guessing, and checking assignments of Boolean variables in the structure of OC nets. As transition systems which are associated with OC nets are rather regular, the method is not straightforward (observe that assignments are easy to handle with a stack; it is not so easy if we only have a counter at our disposal). Using the same technique we also prove that strong bisimilarity between processes of OC nets is **co-NP**-hard (strong bisimilarity between processes of OC automata and finite-state processes is already *polynomial*—see below).

Assuming the expected relationship among complexity classes, the **DP**-hardness result for weak bisimilarity actually says that any deterministic algorithm which decides the problem requires exponential time in the worst case. In Section 3 we design an algorithm which decides weak bisimilarity between a process $p(i)$ of an OC automaton $\mathscr{A}$ and a process f of a finite-state system $\mathscr{F}$ in $\mathcal{O}(n^7 \cdot m^5 \cdot z^5 \cdot (i+1))$ time where n is the size of $\mathscr{A}$, m is the size of $\mathscr{F}$, and z is a special parameter which depends only on $\mathscr{A}$. Note that if there was no z, or if z was always 'small', the

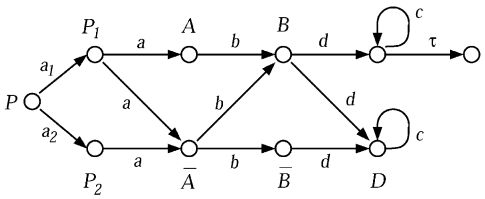


Fig. 1. The finite-state system $\mathscr{F}$ used in the proof of Theorem 7.

problem would be in **P**. In general, z can be exponentially larger than n. However, it follows from the definition of z that the automaton $\mathscr{A}$ must be rather 'special' to make its associated z large (a good example is the automaton constructed in the **DP**-hardness proof of Section 2). Hence, we conclude that our algorithm is actually efficient in a large subclass of instances, and we also give a sort of 'characterization' of hard instances as a byproduct. The algorithm also works for strong bisimilarity, but in this case it only needs polynomial time—we obtain (as a simple corollary) that the problem of strong bisimilarity between OC processes and finite-state ones is in **P**.

As z is the only factor which can blow-up the time complexity of our algorithm, it can be seen as a kind of 'hardness measure'; the time needed to solve a given instance can be estimated by evaluating z. Unfortunately, a straightforward algorithm (which just 'implements' the definition of z) requires exponential time. Therefore, in Section 3.3 we design another parameter Z computable in $\mathcal{O}(n^7)$ time such that $Z \leqslant z < Z \cdot (|Q|+1)$ (where $|Q|$ is the number of control states of $\mathscr{A}$). Thus, the value of z can be efficiently approximated.

Finally, in Section 4 we draw our conclusions and comment on the obtained results.

In the next sections we use $\mathbb{N}$ and $\mathbb{N}_0$ to denote the sets of positive and non-negative integers, respectively.

2. Lower bounds

In this section, we prove that the problem of weak bisimilarity between processes of OC nets and finite-state processes is **DP**-hard and that the problem of strong bisimilarity between processes of OC nets is **co-NP**-hard.

Theorem 7. *The problem of weak bisimilarity between processes of one-counter nets and finite-state processes is* **DP***-hard.*

Proof. For purposes of this proof, we first fix the finite-state system $\mathscr{F}$ of Fig. 1. We show **DP**-hardness by reduction of the **DP**-complete problem SAT–UNSAT. An instance of the SAT–UNSAT problem is a pair (φ_1, φ_2) of Boolean formulae in CNF. The question is whether φ_1 is satisfiable and φ_2 unsatisfiable. First, we describe a polynomial-time algorithm which for a given formula φ in CNF constructs a one-counter net $\mathscr{N}_\varphi$ and

its process $s_\varphi(0)$ such that φ is satisfiable iff $s_\varphi(0) \approx P_1$, and φ is unsatisfiable iff $s_\varphi(0) \approx P_2$, where P_1, P_2 are the (fixed) FS processes of the system $\mathscr{F}$. It clearly suffices for our purposes, because then we can also construct a one-counter net $\mathscr{N}$ by taking the disjoint union of $\mathscr{N}_{\varphi_1}$, $\mathscr{N}_{\varphi_2}$ and adding a new control state s together with transitions $sZ \xrightarrow{a_1} s_{\varphi_1}Z$, $sI \xrightarrow{a_1} s_{\varphi_1}I$ and $sZ \xrightarrow{a_2} s_{\varphi_2}Z$, $sI \xrightarrow{a_2} s_{\varphi_2}I$ (the non-zero transitions are added just to fulfill the constraints of the definition of OC nets). Clearly (φ_1, φ_2) is a positive instance of the SAT–UNSAT problem iff $s(0) \approx P$ where P is the fixed FS process of the system $\mathscr{F}$ (see Fig. 1).

In our proof we use the following theorem of number theory (see, e.g., [4]): Let π_i be the ith prime number, and let $f : \mathbb{N} \to \mathbb{N}$ be a function which assigns to each n the sum $\sum_{i=1}^{n} \pi_i$. Then f is $\mathcal{O}(n^3)$. (In our case, it suffices to know that the sum is asymptotically bounded by a polynomial in n.) With the help of this fact we can readily confirm that the construction described below is indeed polynomial.

Let $\varphi \equiv C_1 \wedge \cdots \wedge C_m$ be a formula in CNF where C_i are clauses over propositional variables $x_1, \ldots, x_n$. We assume (w.l.o.g.) that for every assignment $v : \{x_1, \ldots, x_n\} \to \{\mathtt{true}, \mathtt{false}\}$ there is at least one clause C_i such that $v(C_i) = \mathtt{true}$ (this can be achieved, e.g., by adding the clause $(x_1 \vee \neg x_1)$ to φ). Furthermore, we also assume that φ is not a tautology, i.e., there is at least one assignment v such that $v(\varphi) = \mathtt{false}$ (this just means that there is a clause where no variable appears both positively and negatively). The construction of $\mathscr{N}_\varphi$ will be described in a stepwise manner. First, for each clause C of φ we define the OC net $\mathscr{N}_C = (Q_C, \{I, Z\}, \{c, d, \tau\}, \delta_C)$, where

$$Q_C = \{p, q\} \cup \{\langle j, k\rangle \mid 1 \leqslant j \leqslant n \text{ and } 0 \leqslant k < \pi_j\}$$

consists of $\mathcal{O}(n^4)$ states, and δ_C defines the following transitions:

- $qI \xrightarrow{c} qI$,
- $pI \xrightarrow{d} qI$,
- $pI \xrightarrow{d} \langle j, 0\rangle I$ (for every $1 \leqslant j \leqslant n$),
- $\langle j, k\rangle I \xrightarrow{c} \langle j, k\rangle I$ (for all $1 \leqslant j \leqslant n$ and $0 \leqslant k < \pi_j$),
- $\langle j, k\rangle I \xrightarrow{\tau} \langle j, (k+1) \bmod \pi_j\rangle \varepsilon$ (for all $1 \leqslant j \leqslant n$ and $0 \leqslant k < \pi_j$),
- $\langle j, 0\rangle Z \xrightarrow{c} \langle j, 0\rangle Z$ (for every $1 \leqslant j \leqslant n$) if x_j is not a literal in C,
- $\langle j, k\rangle Z \xrightarrow{c} \langle j, k\rangle Z$ (for all $1 \leqslant j \leqslant n$ and $1 \leqslant k < \pi_j$) if $\neg x_j$ is not a literal in C.

As an example, the transition system generated by $\mathscr{N}_C$, where $C \equiv \neg x_1 \wedge x_3$ and $n = 3$, is shown in Fig. 2. To understand this picture, observe that transition systems associated to OC automata can be viewed as two-dimensional 'tables' where column-indexes are control states and row-indexes are counter values $(0, 1, 2, \ldots)$. As the out-going transitions of a state $q(i)$ where $i > 0$ do not depend on the actual value of i, it suffices to depict the out-going transitions at zero and (some) non-zero level. Note that in Fig. 2, some of the c-loops disappear at zero level. In particular, the state $\langle 1, 1\rangle Z$ has no c-loop because $\neg x_1$ is a literal in C. Similarly, $\langle 3, 0\rangle Z$ has no c-loop because x_3 is a literal in C. Finally, all states $\langle 2, k\rangle Z$ have a c-loop because C does not have literals $x_2, \neg x_2$.

We can then make the following observations:

- c-transitions never change the state.

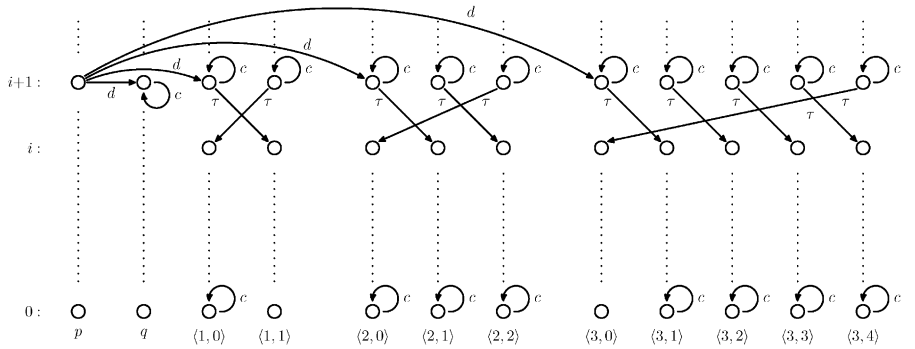


Fig. 2. The structure of $\mathscr{T}_{\mathcal{N}_C}$ for $C = \neg x_1 \vee x_3$, assuming $n = 3$.

- The only 'completed' $\stackrel{d}{\Rightarrow}$-transition sequences from $pI^\ell Z$, that is, those ending in states from which no τ-transition is possible, are
  - $pI^\ell Z \xrightarrow{d} qI^\ell Z$ and
  - $pI^\ell Z \xrightarrow{d} \langle j,0\rangle I^\ell Z \xrightarrow{\tau} \langle j,1\rangle I^{\ell-1} Z \xrightarrow{\tau} \cdots \xrightarrow{\tau} \langle j, \ell \bmod \pi_j\rangle Z$.
- A c-transition is available at every state within the above $\stackrel{d}{\Rightarrow}$-transition sequences after the initial d-transition, except possibly from the last state $\langle j, \ell \bmod \pi_j\rangle Z$. Hence, referring to the system $\mathscr{F}$ of Fig. 1, either
  - $pI^\ell Z \approx \bar{B}$ (if for every $1 \leqslant j \leqslant n$, the state $\langle j, \ell \bmod \pi_j\rangle Z$ has a c-transition); or
  - $pI^\ell Z \approx B$ (otherwise).
- Thus,
  - $pI^\ell Z \approx \bar{B}$ iff for every $1 \leqslant j \leqslant n$:
    if $\pi_j | \ell$ then x_j is not a literal of C; and
    if $\pi_j \nmid \ell$ then $\neg x_j$ is not a literal of C;
  - $pI^\ell Z \approx B$ otherwise; that is, iff for some $1 \leqslant j \leqslant n$:
    $\pi_j | \ell$ and x_j is a literal of C; or
    $\pi_j \nmid \ell$ and $\neg x_j$ is a literal of C.

With these observations in mind, we define the assignment v_ℓ by

$$v_\ell(x_j) = \begin{cases} \texttt{true} & \text{if } \pi_j | \ell, \\ \texttt{false} & \text{otherwise.} \end{cases}$$

Note that any assignment v is equal to v_ℓ where $\ell = \Pi_{1 \leqslant j \leqslant n}\{\pi_j \mid v(x_j) = \texttt{true}\}$. We can then easily verify that

$$v_\ell(C) = \begin{cases} \texttt{true} & \text{if } pI^\ell Z \approx B, \\ \texttt{false} & \text{if } pI^\ell Z \approx \bar{B}. \end{cases}$$

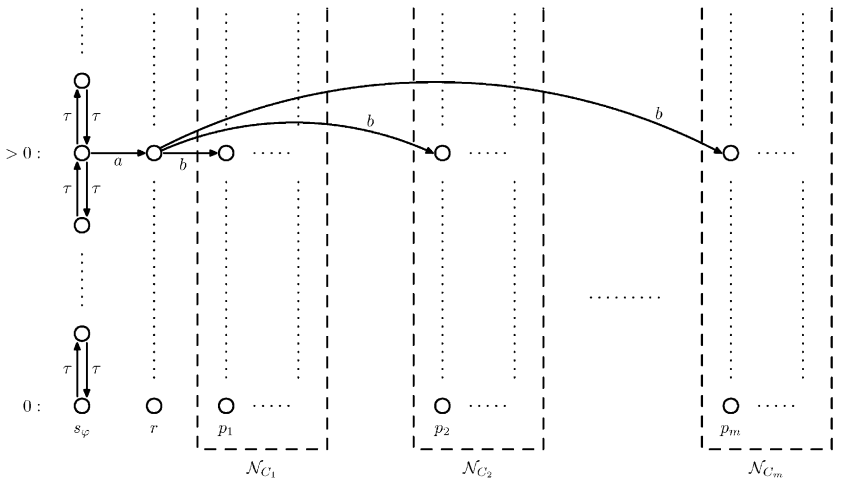


Fig. 3. The structure of $\mathscr{T}_{\mathcal{N}_\varphi}$.

Finally, the OC net $\mathcal{N}_\varphi = (Q, \{I, Z\}, \{a, b, c, d, \tau\}, \delta)$ is defined by taking Q to be the disjoint union of the Q_{C_i} (we shall subscript states associated with clause C_i with the index i, thus getting states p_i, q_i, and $\langle j, k\rangle_i$), along with the two new states s_φ and r; and taking δ to be the disjoint union of the δ_{C_i} along with the following further transitions:

- $s_\varphi Z \xrightarrow{\tau} s_\varphi IZ$,
- $s_\varphi I \xrightarrow{\tau} s_\varphi II$,
- $s_\varphi I \xrightarrow{\tau} s_\varphi \varepsilon$,
- $s_\varphi I \xrightarrow{a} rI$, and
- $rI \xrightarrow{b} p_i I$ (for every $1 \leqslant i \leqslant m$).

(The third rule is not strictly necessary, but it simplifies the proof.) The transition system generated by $\mathcal{N}_\varphi$ is pictured in Fig. 3. We can then make the following sequence of observations.

- $v_\ell(\varphi) = \texttt{true}$

  iff $v_\ell(C_i) = \texttt{true}$ for every $1 \leqslant i \leqslant m$
  iff $p_i I^\ell Z \approx B$ for every $1 \leqslant i \leqslant m$ (by (1) above)
  iff $rI^\ell Z \approx A$.

- $v_\ell(\varphi) = \texttt{false}$

  iff $v_\ell(C_i) = \texttt{false}$ for some $1 \leqslant i \leqslant m$
  iff $p_i I^\ell Z \approx \bar{B}$ for some $1 \leqslant i \leqslant m$ (by (1) above)
  iff $rI^\ell Z \approx \bar{A}$.

For the final step of this argument, we used the assumption that $v_\ell(C_{i'}) = \texttt{true}$ for some clause $C_{i'}$ of φ, so that the transition $\bar{A} \xrightarrow{b} B$ can be matched by $rI^\ell Z \xrightarrow{b} p_{i'} I^\ell Z$.

- φ is unsatisfiable

$$\begin{aligned} &\text{iff } v_\ell(\varphi) = \texttt{false} \text{ for all } \ell \\ &\text{iff } rI^\ell Z \approx \bar{A} \text{ for all } \ell \\ &\text{iff } s_\varphi Z \approx P_2. \end{aligned}$$

- φ is satisfiable

$$\begin{aligned} &\text{iff } v_\ell(\varphi) = \texttt{true} \text{ for some } \ell \\ &\text{iff } rI^\ell Z \approx A \text{ for some } \ell \\ &\text{iff } s_\varphi Z \approx P_1. \end{aligned}$$

For the final step of this argument, we used the assumption that φ is not a tautology, that is, that $v_{\ell'}(\varphi) = \texttt{false}$ for some ℓ', so that the transition $P_1 \xrightarrow{a} \bar{A}$ can be matched by $s_\varphi Z \stackrel{a}{\Rightarrow} rI^{\ell'} Z$. Also note that each transition $s_\varphi Z \stackrel{\tau}{\Rightarrow} s_\varphi I^k Z$ is matched by $P_1 \stackrel{\tau}{\Rightarrow} P_1$, since $s_\varphi Z \stackrel{\tau}{\Leftrightarrow} s_\varphi I^k Z$ so $s_\varphi Z \approx s_\varphi I^k Z$ for all k. □

The main reason why we could not extend the hardness result to some higher complexity class (e.g., **PSPACE**) is that there is no apparent way how to implement a 'stepwise-guessing' of Boolean variables which would allow to encode, e.g., the **PSPACE**-complete quantified Boolean formulae problem; each such attempt resulted in an exponential blow-up in the number of control states.

A natural question is whether the proof of Theorem 7 can be modified so that it also works for *strong* bisimilarity. In the next section we show it is not the case[2]—it turns out that the problem is in **P**. However, we can still re-use our technique to establish a lower bound for the 'symmetric' case:

Theorem 8. *The problem of strong bisimilarity between processes of one-counter nets is* **co-NP**-*hard.*

Proof. We use a similar construction as in the proof of Theorem 7. Given a formula φ in CNF, we construct two one-counter nets $\mathcal{N}, \overline{\mathcal{N}}$ and their processes $s(0), \bar{s}(0)$ such that φ is unsatisfiable iff $s(0) \sim \bar{s}(0)$. The net $\mathcal{N}$ is just a slight modification of the net $\mathcal{N}_\varphi$ of Theorem 7—we only rename all τ-labels to c (in fact, it would suffice to rename those τ transitions which decrease the counter). A key observation is that φ is unsatisfiable iff after each sequence of transitions of the form c^*a (i.e., after each choice of an assignment) there is a b-transition to a state which, after emitting a d-transition, can only emit an infinite sequence of c actions without a possibility to terminate (i.e., at least one clause is false for any assignment). The net $\overline{\mathcal{N}}$ is a 'copy' of $\mathcal{N}$ but we also add new control states u, v and transitions $\bar{r}I \xrightarrow{b} uI$, $uI \xrightarrow{d} vI$, and $vI \xrightarrow{c} vI$, where

[2] Unless **DP** = **P**.

$\bar{r}$ is a 'twin' of the state r of $\mathcal{N}_\varphi$. We put s and $\bar{s}$ to be the corresponding twins of the state s_φ of $\mathcal{N}_\varphi$. Now we can easily check that φ is unsatisfiable iff $s(0) \sim \bar{s}(0)$—the crucial argument is stated above. □

It is worth to note that the technique of Theorem 7 can also be applied to other problems related to formal verification of OC processes. For example, it was used in [30] to show **NP** and **co-NP** hardness of the model-checking problem with (fixed) formulae of the logic EF and processes of OC nets; in the same paper, it was also shown that *simulation* equivalence between processes of OC automata and finite-state processes is **co-NP**-hard.

3. Upper bounds

Since the problem of weak bisimilarity between processes of OC nets and finite-state processes is **DP**-hard (Theorem 7), it is likely that any deterministic algorithm solving the problem requires exponential time in the worst case. Existing algorithms for checking weak bisimilarity between PDA and finite-state processes have this complexity, and can be thus seen as 'essentially time-optimal'. These algorithms are usually based on reductions to the model-checking problem—for example, given a PDA process P and a finite-state process F, one can construct a *characteristic formula* [45,38] for F in the modal μ-calculus and then decide whether P satisfies the formula. A characteristic formula for F can also be constructed in the logic EF [25] (more precisely, in a slightly extended version of EF which can also impose restrictions on sequences of atomic actions; even this extended logic still forms a simple fragment of the modal μ-calculus). This 'model-checking approach' has two disadvantages. First, it does not give any idea on what actually makes the problem of bisimilarity-checking between OC and finite-state processes hard. Second, it results in an exponential-time algorithm even in the case of *strong* bisimilarity. It seems to be inevitable for PDA (strong bisimilarity between PDA and finite-state processes is **PSPACE**-hard [35]), but there is no lower bound for the problem of strong bisimilarity between OC and finite-state processes (as we shall see, the problem is actually in **P**). As model-checking with EF is both **NP** and **co-NP** hard for processes of OC nets [30], the model-checking approach does not yield an efficient algorithm for checking strong bisimilarity between OC and finite-state processes.

In this section we design a new algorithm which decides weak bisimilarity between processes of OC automata and finite-state processes. First, let us fix

- a one-counter automaton $\mathscr{A} = (Q, \{I, Z\}, \Sigma, \delta)$ of size n,
- a finite-state system $\mathscr{F} = (F, \Sigma, \rightarrow)$ of size m.

As expected, our algorithm requires exponential time in the worst case. More precisely, it needs $\mathcal{O}(n^7 \cdot m^5 \cdot z^5 \cdot (i+1))$ time[3] to decide weak bisimilarity between processes $p(i)$ of $\mathscr{A}$ and f of $\mathscr{F}$. Here, the 'z' is a special parameter which depends only on $\mathscr{A}$

[3] Note that we need a non-constant time even in the particular case when $i = 0$ (the problem is still **DP**-hard). That is why we write '$i+1$'.

(and which can be exponentially larger than n). To motivate the following technical development, we summarize the main features of our algorithm and outcomes of the underlying analysis.

- In the case of *strong* bisimilarity, the parameter z equals 1; this means that our algorithm decides strong bisimilarity between processes of OC automata and finite-state ones in *polynomial* time.
- In the case of weak bisimilarity, it is only the size of z which can make the problem computationally hard. However, it follows from the definition of z that $\mathscr{A}$ must be rather 'special' to make its associated z large. Hence,
  - the algorithm works efficiently in a large subclass of instances;
  - we obtain a sort of 'characterization' of hard instances of the problem which also suggests that the proof of Theorem 7 cannot be much simplified.
- Although the parameter z can be effectively computed from the structure of $\mathscr{A}$, its computation can take time exponential in n. Nevertheless, we give an algorithm which computes a (usually tight) upper approximation Z of z in *polynomial* time. This means that we can efficiently compute a sort of 'hardness measure' whose small value guarantees fast termination of our algorithm.

3.1. Auxiliary results

We start by recalling some notions and results which will be later used in our constructions. To make this paper self-contained, we also sketch crucial proofs.

Let $\mathscr{T} = (S, \Sigma, \to)$ be a transition system. For each $i \in \mathbb{N}_0$ we define the relation $\approx_i \subseteq S \times S$ inductively as follows:

- $\approx_0 \; = S \times S$,
- $s \approx_{i+1} t$ iff for each $s \stackrel{a}{\Rightarrow} s'$ there is some $t \stackrel{a}{\Rightarrow} t'$ such that $s' \approx_i t'$, and vice versa.

It is easy to check that each $\approx_i$ is an equivalence relation. We also use $\approx_i$ to relate states of different transition systems; formally, we consider two transition systems to be a single one by taking their disjoint union. The following theorem has been established in [25] (see also [28,2,24]):

Theorem 9. *Let $\mathscr{G} = (G, \Sigma, \to)$ be a (general) transition system and $\mathscr{F} = (F, \Sigma, \to)$ a finite-state system. We say that a state $g \in G$ is i-*good *for a given $i \in \mathbb{N}_0$ iff there is $f \in F$ such that $g \approx_i f$; g is i-*bad *iff g is not i-good.*

Let $k \in \mathbb{N}$ be greater or equal to the number of states of $\mathscr{F}$. Let $g \in G$ and $f \in F$. It holds that $g \approx f$ iff $g \approx_k f$ and each state which is reachable from g is k-good.

Proof. First realize that since $\approx_i$ is an equivalence relation and $\approx_{i+1}$ refines $\approx_i$ for each $i \in \mathbb{N}_0$, the quotient of F under $\approx_k$ is the same as the quotient of F under $\approx_{k-1}$. In other words, for all $f, f' \in F$ we have that $f \approx_k f'$ iff $f \approx_{k-1} f'$.

Now let $g \in G$ and $f \in F$ be two states such that $g \approx_k f$ and each state which is reachable from g is k-good. We show that the relation

$$\mathscr{R} = \{(g', f') \mid g \to^* g', f' \in F, g' \approx_k f'\}.$$

is a weak bisimulation. Let $(g', f') \in \mathscr{R}$. By definition of $\approx_k$, for each move $g' \stackrel{a}{\Rightarrow} g''$ there is a move $f' \stackrel{a}{\Rightarrow} f''$ such that $g'' \approx_{k-1} f''$ (and vice versa). As g'' is reachable from g', it is also reachable from g and hence it is k-good—there is some $\overline{f} \in F$ such that $g'' \approx_k \overline{f}$. By transitivity of $\approx_{k-1}$ we now obtain $f'' \approx_{k-1} \overline{f}$, hence also $f'' \approx_k \overline{f}$ due to the observation above. Now we use transitivity of $\approx_k$ to conclude that $g'' \approx_k f''$, hence $(g'', f'') \in \mathscr{R}$ as required. □

A non-deterministic finite automaton (NFA) is formally understood as a tuple $\mathscr{M} = (S, \Sigma, \hookrightarrow, F)$ where $(S, \Sigma, \hookrightarrow)$ is a finite-state transition system and $F \subseteq S$ a set of accepting states. For each $s \in S$ we define its language

$$L(s) = \{w \in \Sigma^* \mid \exists f \in F : s \stackrel{w}{\rightarrow} f\}.$$

A proof of the next theorem can be found in [3].

Theorem 10. *Let $\mathscr{M} = (S, \Sigma, \hookrightarrow, F)$ be a NFA, $w \in \Sigma^*$, and $s \in S$. The problem if $w \in L(s)$ is decidable in $\mathcal{O}(|w| \cdot |S|^2)$ time.*

To be able to represent infinite sets of OC processes in a finite and compact way, we borrow the following 'tool' from [8]:

Definition 11. Let $\mathscr{P} = (Q, \Gamma, \Sigma, \delta)$ be a pushdown automaton, $\mathscr{M} = (S, \Gamma, \hookrightarrow, F)$ a NFA (note that the set of actions of $\mathscr{M}$ is the stack alphabet of $\mathscr{P}$), and $Init : Q \to S$ a total injective function. A process $p\alpha$ of $\mathscr{P}$ is recognized by the pair $(\mathscr{M}, Init)$ iff $\alpha \in L(Init(p))$.

The next theorem also is taken from [8] (our presentation and complexity analysis is actually based mainly on [7] where the same problem is considered in the framework of context-free grammars).

Theorem 12. *Let $\mathscr{P} = (Q, \Gamma, \Sigma, \delta)$ be a pushdown automaton, $\mathscr{M} = (S, \Gamma, \hookrightarrow, F)$ a NFA, and $Init : Q \to S$ a total injective function. Let N be the set of processes recognized by $(\mathscr{M}, Init)$. Then one can effectively construct an automaton $\mathscr{M}' = (S, \Gamma, \rightsquigarrow, F)$ in time $\mathcal{O}(|\Gamma| \cdot |\delta| \cdot |S|^5)$ such that $(\mathscr{M}', Init)$ recognizes the set*

$$Pre^*(N) = \{q\beta \mid q\beta \rightarrow^* p\alpha \ \textit{for some} \ p\alpha \in N\}$$

of all predecessors of N.

Proof (Sketch). The transition relation $\rightsquigarrow$ of $\mathscr{M}'$ can be computed, e.g., by the algorithm given in Fig. 4. We refer to [8] for a correctness proof. Let us evaluate its complexity. As $\rightsquigarrow$ cannot have more than $|\Gamma| \cdot |S|^2$ elements, the **for** loop is executed $\mathcal{O}(|\Gamma| \cdot |S|^2)$ times. Each time, the **if** command is executed $\mathcal{O}(|\delta| \cdot |S|)$ times. The condition whether $q \stackrel{\alpha}{\rightsquigarrow} r$ can be verified in $\mathcal{O}(|S|^2)$ time (see Theorem 10 and realize that $|\alpha| \leqslant 2$), and possible updating of $\rightsquigarrow$ can be done in constant time (if $\rightsquigarrow$ is stored as, e.g., a bit matrix). Hence, we need $\mathcal{O}(|\Gamma| \cdot |\delta| \cdot |S|^5)$ time in total. □

Input: $\mathscr{P}=(Q,\Gamma,\Sigma,\delta)$, $\mathscr{M}=(S,\Gamma,\hookrightarrow,F)$, $Init$
Output: $\rightsquigarrow$
$\rightsquigarrow := \hookrightarrow$;
repeat
  for all $pX \xrightarrow{a} q\alpha \in \delta$, $r \in S$ **do**
    if $Init(q) \overset{\alpha}{\rightsquigarrow} r$ **then** $\rightsquigarrow := \rightsquigarrow \cup \{(Init(p),X,r)\}$ **fi**
  od
until $\rightsquigarrow$ does not change anymore

Fig. 4. An algorithm for computing $\mathscr{M}'$.

The algorithm of Fig. 4 is rather simple and inefficient. A more careful implementation of the same idea can result in better algorithms (see, e.g., [17,7]). However, it is not so important in our setting; the main point is that $\mathscr{M}'$ is constructible in polynomial time.

3.2. The algorithm

Intuition: To decide weak bisimilarity between processes $p(i)$ of $\mathscr{A}$ and f of $\mathscr{F}$, it suffices (by Theorem 9) to find out if $p(i) \approx_m f$ and whether $p(i)$ can reach a state which is m-bad. We do that by constructing a constant z such that for each state $q(j)$ of $\mathscr{T}_{\mathscr{A}}$ where $j \geqslant 4(m+1)z$ we have that $q(j) \approx_m q(j-z)$. In other words, each state of $\mathscr{T}_{\mathscr{A}}$ is (up to $\approx_m$) represented by another (and effectively constructible) state whose counter value is bounded by $4(m+1)z$. Then we convert this 'initial part' of $\mathscr{T}_{\mathscr{A}}$ to a finite-state system $\mathscr{F}_{\mathscr{A}}$ and construct the $\approx_m$ relation between the states of $\mathscr{F}_{\mathscr{A}}$ and $\mathscr{F}$. The question if $p(i) \approx_m f$ is then easy to answer (we look if the representative of $p(i)$ within $\mathscr{F}_{\mathscr{A}}$ is related to f by $\approx_m$). The question if $p(i)$ can reach a state which is m-bad still requires some development—we observe that states which are m-bad are 'regularly distributed' over $\mathscr{T}_{\mathscr{A}}$ and construct a description of that distribution (which is 'read' from $\mathscr{F}_{\mathscr{A}}$) in a form of a NFA $\mathscr{M}$ and a function $Init$ such that $(\mathscr{M},Init)$ recognizes all m-bad states. Then we use the algorithm of Fig. 4 to construct a NFA $\mathscr{M}'$ such that $(\mathscr{M}',Init)$ recognizes the set of all states which can reach a state recognized by $(\mathscr{M},Init)$, and look whether $(\mathscr{M}',Init)$ recognizes $p(i)$. All procedures we use are polynomial in the size of $\mathscr{F}_{\mathscr{A}}$. Hence, it is only the size of z which can blow-up the time complexity.

The next definitions and lemmata reveal a crucial periodicity in the structure of $\mathscr{T}_{\mathscr{A}}$.

Definition 13. For all $a \in \Sigma$ and $l \in \mathbb{N}_0$ we define a binary relation $\overset{a}{\Rightarrow}_l$ over the set of states of $\mathscr{T}_{\mathscr{A}}$ as follows: $p(i) \overset{a}{\Rightarrow}_l q(j)$ iff there is a sequence of transitions from $p(i)$ to $q(j)$ which forms a '$\overset{a}{\Rightarrow}$' move and the counter value remains greater than or equal to l in all states which appear in the sequence (including $p(i)$ and $q(j)$).

At the core of our analysis are observations about the structure of $\stackrel{\tau}{\Rightarrow}_l$ relations. We start with a simple one, which is an immediate consequence of the aforementioned 'regularity' of $\mathscr{T}_{\mathscr{A}}$.

Lemma 14. *For all $i,j \in \mathbb{N}$ and $p,q \in Q$ we have that $p(i+j) \stackrel{\tau}{\rightarrow}_i q(i)$ iff $p(1+j) \stackrel{\tau}{\rightarrow}_1 q(1)$.*

To simplify our notation, we introduce another family of relations.

Definition 15. For each $j \in \mathbb{N}$ we define a relation $\hookrightarrow_j \subseteq Q \times Q$ as follows: $p \hookrightarrow_j q$ iff $p(1+j) \stackrel{\tau}{\Rightarrow}_1 q(1)$.

Due to Lemma 14 we immediately see that $p(i+j) \stackrel{\tau}{\Rightarrow}_i q(i)$ iff $p \hookrightarrow_j q$.

Lemma 16. *For each $j \in \mathbb{N}$ we have $\hookrightarrow_j = \underbrace{\hookrightarrow_1 \circ \cdots \circ \hookrightarrow_1}_{j}$.*

Proof. By induction on j. The base case ($j=1$) is immediate. Now let $p \hookrightarrow_{j+1} q$. Then $p(2+j) \stackrel{\tau}{\Rightarrow}_1 q(1)$ (by Definition 15), hence there is a sequence of τ transitions from $p(2+j)$ to $q(1)$ which never decreases the counter below 1. Let $r(1+j)$ be the first state of the sequence where the counter is decreased to $1+j$. We see that $p(2+j) \stackrel{\tau}{\rightarrow}_{1+j} r(1+j)$, hence $p(2) \stackrel{\tau}{\Rightarrow}_1 r(1)$ due to Lemma 14 and $p \hookrightarrow_1 r$ by Definition 15. Furthermore, $r(1+j) \stackrel{\tau}{\Rightarrow}_1 q(1)$, hence $r \hookrightarrow_j q$ (again by Definition 15). To sum up, we obtain $\hookrightarrow_{j+1} = \hookrightarrow_j \circ \hookrightarrow_1$ and we can apply the induction hypotheses to finish the proof. □

Definition 17. For each $p \in Q$ we define its *characteristic sequence* $\mathscr{C}_p : \mathbb{N}_0 \to 2^Q$ as follows:
- $\mathscr{C}_p(0) = \{p\}$;
- $\mathscr{C}_p(i+1) = \{q \in Q \mid \exists r \in \mathscr{C}_p(i)$ such that $r \hookrightarrow_1 q\}$.

The next lemma is an immediate consequence of Definition 17 and Lemma 16.

Lemma 18. *For all $p \in Q$, $i \in \mathbb{N}$, and $j \in \mathbb{N}_0$ we have that $q \in \mathscr{C}_p(j)$ iff $p(i+j) \stackrel{\tau}{\Rightarrow}_i q(i)$.*

Another simple observation is that the sequence $\mathscr{C}_p$ is (for every $p \in Q$) ultimately periodic—as 2^Q is a finite set, there is $i \in \mathbb{N}_0$ such that $\mathscr{C}_p(i) = \mathscr{C}_p(j)$ for some $j > i$; let us choose the smallest such i and for this i the smallest j. Now we put
- $\alpha_p = \mathscr{C}_p(0) \cdots \mathscr{C}_p(i-1)$,
- $\beta_p = \mathscr{C}_p(i) \cdots \mathscr{C}_p(j-1)$.

Hence, α_p can be empty while β_p is always non-empty; it can also happen that β_p consists of just one element $\emptyset$. Also observe that the sets contained in α_p, β_p are pairwise different, because otherwise we would immediately obtain a contradiction to the minimality of i and j. Moreover, $\mathscr{C}_p = \alpha_p \beta_p^\omega$ because $\mathscr{C}_p(i+1)$ is completely determined by $\mathscr{C}_p(i)$.

Definition 19. For each $p \in Q$ we define the *prefix* and *period* of p, denoted $pre(p)$ and $per(p)$, to be the length of α_p and β_p, respectively. Furthermore, we put

$$z = \max\{pre(p) \mid p \in Q\} \cdot \mathrm{lcm}\{per(p) \mid p \in Q\},$$

where $\mathrm{lcm}(M)$ denotes the least common multiple of the elements of M.

Remark 20 (A characterization of hard instances). As we shall see (Lemma 32), each $pre(p)$ is $\mathcal{O}(n^2)$. This means that

$$\mathrm{lcm}\{per(p) \mid p \in Q\}$$

is the only factor which can blow up the size of z and hence also the time complexity of our algorithm for checking weak bisimilarity between processes of OC automata and finite-state processes. To see that $\mathrm{lcm}\{per(p) \mid p \in Q\}$ can *indeed* be exponentially larger than n, it suffices to examine the net $\mathcal{N}_\varphi$ constructed in the proof of Theorem 7. Thus, we obtain a kind of 'characterization' of hard instances of the problem. Intuitively, OC automata presented in hard instances must contain many 'decreasing τ-cycles' of an incomparable length. It also indicates that the 'trick' with prime numbers used in the proof of Theorem 7 is in some sense inevitable.

As $z \geqslant pre(p)$ and $per(p)$ divides z for each $p \in Q$, we obtain the following:

Lemma 21. *For all $p \in Q$ and $i \geqslant z$ we have that $\mathscr{C}_p(i) = \mathscr{C}_p(i+z)$.*

Possible non-bisimilarity of states of the form $p(j)$, $p(j+z)$ (where $j > 0$) cannot be demonstrated without decreasing the counter to zero at some point (as long as the counter remains positive, each of the two processes can just 'mimick' the moves of the other process by performing the same operation on the counter). However, the counter should not be decreased 'too much' in a single $\stackrel{a}{\Rightarrow}$ transition, as it is shown in the next two lemmata.

Lemma 22. *For all $p \in Q$ and $j \in \mathbb{N}$ it holds that*
- *if there is a sequence of τ-transitions from $p(j+2z)$ to (some) $q(l)$ which decreases the counter to j at some point, then $p(j+z) \stackrel{\tau}{\Rightarrow} q(l)$;*
- *if there is a sequence of τ-transitions from $p(j+z)$ to (some) $q(l)$ which decreases the counter to j at some point, then $p(j+2z) \stackrel{\tau}{\Rightarrow} q(l)$.*

Proof. We show only the first part (the other one is similar). As there is a sequence of τ-transitions from $p(j+2z)$ to $q(l)$ which decreases the counter to j, there must be an intermediate state $r(j)$ such that $p(j+2z) \stackrel{\tau}{\rightarrow}_j r(j) \stackrel{\tau}{\Rightarrow} q(l)$. As $p(j+2z) \stackrel{\tau}{\rightarrow}_j r(j)$, we obtain that $r \in \mathscr{C}_p(2z)$ due to Lemma 18, hence also $r \in \mathscr{C}_p(z)$ by Lemma 21. From this we have (again by Lemma 18) that $p(j+z) \stackrel{\tau}{\rightarrow}_j r(j)$, hence $p(j+z) \stackrel{\tau}{\Rightarrow} q(l)$ as required. □

Lemma 23. *For all $p \in Q$ and $j \in \mathbb{N}$ it holds that*

- *if there is a sequence of transitions forming one '$\stackrel{a}{\Rightarrow}$' move from $p(j+4z)$ to (some) $q(l)$ which decreases the counter to j at some point, then $p(j+3z) \stackrel{a}{\Rightarrow} q(l)$;*
- *if there is a sequence of transitions forming one '$\stackrel{a}{\Rightarrow}$' move from $p(j+3z)$ to (some) $q(l)$ which decreases the counter to j at some point, then $p(j+4z) \stackrel{a}{\Rightarrow} q(l)$.*

Proof. Again, we only show the first part. The case when $a = \tau$ has been handled in Lemma 22. If $a \neq \tau$, we can distinguish two cases:

- The counter is decreased to $j+2z$ at some point in the considered sequence of transitions before emitting the action a. Then there is a state $r(j+2z)$ such that $p(j+4z) \stackrel{\tau}{\Rightarrow} r(j+2z) \stackrel{a}{\Rightarrow} q(l)$. Now we can apply Lemma 22 and conclude that $p(j+3z) \stackrel{\tau}{\Rightarrow} r(j+2z)$ which suffices.
- The action a is emitted before decreasing the counter to $j+2z$. Then there is a state $r(j+2z)$ such that $p(j+4z) \stackrel{a}{\rightarrow}_{j+2z} r(j+2z)$ and $r(j+2z)$ enters the state $q(l)$ in a sequence of τ-transitions which decreases the counter to j. Hence, by Lemma 22 we know that $r(j+z) \stackrel{\tau}{\Rightarrow} q(l)$. Now it suffices to realize that since $p(j+4z) \stackrel{a}{\rightarrow}_{j+2z} r(j+2z)$, we also have $p(j+3z) \stackrel{a}{\rightarrow}_{j+z} r(j+z)$. To sum up, $p(j+3z) \stackrel{a}{\rightarrow}_{j+z} r(j+z) \stackrel{\tau}{\Rightarrow} q(l)$ and we are done. □

Lemma 24. *Let $p \in Q$ and $k \in \mathbb{N}_0$. For each $c > 4(k+1)z$ we have that $p(c) \approx_k p(c-z)$.*

Proof. By induction on k. The base case ($k=0$) is immediate. Now let $c > 4(k+2)z$. We prove that for each '$\stackrel{a}{\Rightarrow}$' move of $p(c)$ there is a '$\stackrel{a}{\Rightarrow}$' move of $p(c-z)$ such that the resulting pair of states is related by $\approx_k$, and vice versa. Let $p(c) \stackrel{a}{\Rightarrow} q(l)$. We distinguish two cases:

- $p(c) \stackrel{a}{\rightarrow}_{c-4z+1} q(l)$. This means that $l \geqslant c-4z+1$, hence $l > 4(k+1)z$. Furthermore, we have $p(c-z) \stackrel{a}{\rightarrow}_{c-5z+1} q(l-z)$ (this move is possible because $c > 5z$). Now $q(l) \approx_k q(l-z)$ by induction hypotheses.
- The counter is decreased to $c-4z$ by the considered sequence of transitions. Then $p(c-z) \stackrel{a}{\Rightarrow} q(l)$ by Lemma 23. Clearly $q(l) \approx_k q(l)$.

The other direction is shown in a similar way. □

The next lemma presents a part of our complexity analysis.

Lemma 25. *Let $j \in \mathbb{N}$. The first j elements of all characteristic sequences can be computed in $\mathcal{O}(n^7 + j \cdot n^3)$ time.*

Proof. First we show that the $\hookrightarrow_1$ relation can be computed in $\mathcal{O}(n^7)$ time. To do that, we construct a pushdown automaton $\mathscr{P}$ by 'cutting-off' all zero transitions and all 'non-τ' transitions of $\mathscr{A}$. Formally, $\mathscr{P} = (Q, \{I\}, \{\tau\}, \gamma)$ where $pI \stackrel{\tau}{\rightarrow} qI^k \in \gamma$ iff $pI \stackrel{\tau}{\rightarrow} qI^k \in \delta$. We see that $p \hookrightarrow_1 q$ iff $pII \rightarrow^* qI$ in $\mathscr{T}_{\mathscr{P}}$. For each of $\mathcal{O}(n)$ states $q \in Q$ we now construct a NFA $\mathscr{M}_q$ and a function $Init_q$ such that the pair $(\mathscr{M}_q, Init_q)$ recognizes exactly the singleton $\{qI\}$. Observe that $\mathscr{M}_q$ has $|Q|+1$ states (remember that $Init_q$ is injective) and one transition; hence, its size is $\mathcal{O}(n)$. Now we compute the automaton $\mathscr{M}'_q$

of Theorem 12, which takes $\mathcal{O}(n^6)$ time. For each $p \in Q$ we now check if $(\mathscr{M}'_q, Init_q)$ recognizes pII. This can be done (for a given p) in $\mathcal{O}(n^2)$ time (see Theorem 10), hence we need $\mathcal{O}(n^3)$ time for all control states, which is dominated by $\mathcal{O}(n^6)$. To sum up, $\mathcal{O}(n^7)$ time suffices for computation of $\hookrightarrow_1$.

Since

$$\mathscr{C}_p(i+1) = \{q \in Q \mid \exists r \in \mathscr{C}_p(i) \text{ such that } r \hookrightarrow_1 q\}$$

by definition, we only need $\mathcal{O}(n^2)$ time to compute $\mathscr{C}_p(i+1)$ from $\mathscr{C}_p(i)$ and $\hookrightarrow_1$, which gives us the bound $\mathcal{O}(j \cdot n^2)$ for computation of the first j elements of $\mathscr{C}_p$. As there are $\mathcal{O}(n)$ control states, we need $\mathcal{O}(n^7 + j \cdot n^3)$ time in total. □

The following lemma says how much time is needed to compute the constant z. At first glance, it might look strange that the presented time bound for computing z itself depends on z. Nevertheless, it does make sense because the computation of z is a part of our algorithm for deciding weak bisimilarity between processes of OC automata and finite-state processes. One could also easily show that z can be computed in time which is *exponential* in n; however, our aim is to show that the whole algorithm (and hence also the procedure which computes z) is *polynomial* in n, m, and z. The issue is addressed in greater detail in Section 3.3.

Lemma 26. *The constant z is computable in $\mathcal{O}(n^7 + z \cdot n^3)$ time.*

Proof. By definition of z, we first need to compute $pre(p)$ and $per(p)$ for each $p \in Q$. As $pre(p) + per(p)$ is bounded by z, it suffices to compute the first z elements of each C_p which takes $\mathcal{O}(n^7 + z \cdot n^3)$ time by Lemma 25. Now we have to select the maximal $pre(p)$ and multiply it by the least common multiple of all $per(p)$. The required arithmetic can be (comfortably) performed in $\mathcal{O}(n^7 + z \cdot n^3)$ time. □

Now we are in position to prove the main theorem of this section.

Theorem 27. *The problem of weak bisimilarity between processes $p(i)$ of $\mathscr{A}$ and f of $\mathscr{F}$ is decidable in $\mathcal{O}(n^7 \cdot m^5 \cdot z^5 \cdot (i+1))$ time.*

Proof. By Theorem 9, we need to find out whether $p(i) \approx_m f$ and whether $p(i)$ can reach a state which is m-bad. Due to Lemma 24 we know that the set

$$\{p(i) \mid p \in Q, 0 \leqslant i \leqslant 4(m+1)z\}$$

represents the whole state-space of $\mathscr{T}_{\mathscr{A}}$ up to $\approx_m$. Formally, we first define the function $\mathscr{B}$ over the states of $\mathscr{T}_{\mathscr{A}}$ as follows:

$$\mathscr{B}(q(j)) = \begin{cases} q(j) & \text{if } j \leqslant 4(m+1)z; \\ q(4(m+1)z) & \text{if } j > 4(m+1)z \text{ and } (j \bmod z) = 0; \\ q(4mz + 3z + (j \bmod z)) & \text{if } j > 4(m+1)z \text{ and } (j \bmod z) \neq 0. \end{cases}$$

An immediate consequence of Lemma 24 is that $q(j) \approx_m \mathscr{B}(q(j))$ for all $q \in Q$ and $j \in \mathbb{N}_0$. Now we define a finite-state system $\mathscr{F}_{\mathscr{A}} = (F_{\mathscr{A}}, \Sigma, \hookrightarrow)$ where

- $F_{\mathscr{A}}$ is the image of $\mathscr{B}$, i.e., $F_{\mathscr{A}} = \{q(j) \mid q \in Q, 0 \leqslant j \leqslant 4(m+1)z\}$,
- Σ is the set of actions of $\mathscr{A}$,
- $\hookrightarrow$ is the least relation satisfying the following: if $r(k) \xrightarrow{a} s(l)$ is a transition of $\mathscr{T}_{\mathscr{A}}$, then $\mathscr{B}(r(k)) \overset{a}{\hookrightarrow} \mathscr{B}(s(l))$.

The definition of $\mathscr{F}_{\mathscr{A}}$ is effective; however, note that we also have to compute the constant z which takes $\mathcal{O}(n^7 + z \cdot n^3)$ time by Lemma 26. Furthermore, observe that $\mathscr{F}_{\mathscr{A}}$ is actually the 'initial part' of $\mathscr{T}_{\mathscr{A}}$. The only difference is that all up-going transitions of states at level $4(m+1)z$ are 'bent' down to the corresponding $\approx_m$-equivalent states at level $4mz + 3z + 1$. Note that for each $q(j)$ we still have that $q(j) \approx_m \mathscr{B}(q(j))$ when $\mathscr{B}(q(j))$ is seen as a state of $\mathscr{F}_{\mathscr{A}}$. The number of states of $\mathscr{F}_{\mathscr{A}}$ is $\mathcal{O}(n \cdot m \cdot z)$; moreover, as the number of out-going transitions at each 'level' of $\mathscr{T}_{\mathscr{A}}$ is $\mathcal{O}(n)$, the size of $\hookrightarrow$ is $\mathcal{O}(n \cdot m \cdot z)$, which means that the total size of $\mathscr{F}_{\mathscr{A}}$ is also $\mathcal{O}(n \cdot m \cdot z)$.

Now, let us realize that if we have a finite-state system $\mathscr{T}$ of size t (which means that $\mathscr{T}$ has at most t states and transitions), then $\mathcal{O}(t^3)$ time suffices to compute all sets $reach(s,a) = \{r \mid s \overset{a}{\Rightarrow} r\}$ where s is a state and a an action of $\mathscr{T}$—each $reach(s,\tau)$ can be constructed in $\mathcal{O}(t)$ time, hence we need $\mathcal{O}(t^2)$ time to construct all of them. Then, for all states s,t and each transition $u \xrightarrow{a} v$ we check if $u \in reach(s,\tau)$, $t \in reach(v,\tau)$ (these conditions can be verified in constant time now) and if so, we add t to $reach(s,a)$. Therefore, we need $\mathcal{O}(n^3 \cdot m^3 \cdot z^3)$ time to construct the *reach* sets for $\mathscr{F}_{\mathscr{A}}$ and $\mathscr{F}$.

To compute the $\approx_m$ relation between the states of $\mathscr{F}_A$ and $\mathscr{F}$, we define

- $\mathscr{R}^0 = F_{\mathscr{A}} \times F$,
- $\mathscr{R}^{i+1} = Exp(\mathscr{R}^i)$,

where the function $Exp : (F_{\mathscr{A}} \times F) \to (F_{\mathscr{A}} \times F)$ refines its argument according to the definition of $\approx_i$—a pair $(r(j), g)$ belongs to $Exp(\mathscr{R})$ iff it belongs to $\mathscr{R}$ and for each '$\overset{a}{\Rightarrow}$' move of one component there is a corresponding '$\overset{a}{\Rightarrow}$' move of the other component such that the resulting pair of states belongs to $\mathscr{R}$. Clearly, for each pair $(r(j), g)$ of $F_{\mathscr{A}} \times F$ we have that $r(j) \approx_m g$ iff $(r(j), g) \in \mathscr{R}^m$. It remains to clarify the time costs. The function Exp is computed m times. Each time, $\mathcal{O}(n \cdot m^2 \cdot z)$ pairs are examined. For each such pair we have to check the membership to $Exp(\mathscr{R})$. This takes only $\mathcal{O}(n \cdot m^2 \cdot z)$ time, because the extended transition relations of $\mathscr{F}_{\mathscr{A}}$ and $\mathscr{F}$ have already been computed (for each of $\mathcal{O}(n \cdot m \cdot z)$ successors of the first component we try out $\mathcal{O}(m)$ successors of the second components, and vice versa) . To sum up, we need $\mathcal{O}(n^3 \cdot m^5 \cdot z^3)$ time in total.

To check if $p(i) \approx_m f$, we simply look if $(\mathscr{B}(p(i), f)) \in \mathscr{R}^m$. It remains to find out whether $p(i)$ can reach a state $q(j)$ which is m-bad. Observe that $q(j)$ is m-bad iff the state $\mathscr{B}(q(j))$ of $\mathscr{F}_{\mathscr{A}}$ is m-bad. Therefore, we can easily construct a NFA $\mathscr{M}$ and a function *Init* such that the pair $(\mathscr{M}, Init)$ recognizes the set of all m-bad states of $\mathscr{T}_{\mathscr{A}}$—we put $\mathscr{M} = (S, \{I, Z\}, \hookrightarrow, \{fin\})$ where

$$S = \{fin\} \cup \{p(i) \mid p \in Q, 0 \leqslant i \leqslant 4(m+1)z\}$$

and $\hookrightarrow$ is the least transition relation satisfying the following:

- $p(i) \overset{I}{\hookrightarrow} p(i+1)$ for all $p \in Q$, $0 \leqslant i < 4(m+1)z$;

- $p(4(m+1)z) \overset{I}{\hookrightarrow} p(4mz+3z+1)$ for each $p \in Q$;
- if a state $p(i)$ of $\mathscr{F}_{\mathscr{A}}$ is m-bad, then $p(i) \overset{Z}{\hookrightarrow} fin$.

The function *Init* is defined by $Init(p) = p(0)$ for all $p \in Q$. Note that $\mathscr{M}$ has $\mathcal{O}(n \cdot m \cdot z)$ states. Now we compute the automaton $\mathscr{M}'$ of Theorem 12 (it takes $\mathcal{O}(n^6 \cdot m^5 \cdot z^5)$ time) and check if $(\mathscr{M}', Init)$ recognizes $p(i)$. This can be done in $\mathcal{O}(n^2 \cdot m^2 \cdot z^2 \cdot (i+1))$ time because $\mathscr{M}'$ has the same set of states as $\mathscr{M}$ (see Theorem 10).

We see that $\mathcal{O}(n^7 \cdot m^5 \cdot z^5 \cdot (i+1))$ time suffices for all of the aforementioned procedures (the 'n^7' factor comes from the bound for computation of z). □

Our algorithm can also be used to decide *strong* bisimilarity between $p(i)$ and f—we just rename all τ-transitions of $\mathscr{A}$ and $\mathscr{F}$ with some (fresh) action e (it does not change anything from the point of view of strong bisimilarity, because here the τ-transitions are treated as 'ordinary' ones). As there are no τ-transitions anymore, there is no difference between strong and weak bisimilarity, hence we can use the designed algorithm. Since there are no τ's, z equals 1, and so we can conclude:

Corollary 28. *The problem of strong bisimilarity between processes $p(i)$ of $\mathscr{A}$ and f of $\mathscr{F}$ is in* **P**.

3.3. Computing a hardness measure

We have seen (Lemma 26) that the constant z can be computed in $\mathcal{O}(n^7 + z \cdot n^3)$ time. If we only used z in our algorithm for deciding weak bisimilarity between OC and finite-state processes, there would be no need to compute it more efficiently because the algorithm would remain quite time-consuming anyway. However, by computing z one can also estimate the time which is needed to solve a given instance (z can be thus seen as a kind of 'hardness measure'—if it is small, then we get the answer quickly). It is therefore reasonable to ask whether we could determine the value of z in time which is *polynomial in n* (i.e., without constructing the first z elements of every $\mathscr{C}_p$).

Remember that $z = \max\{pre(p) \mid p \in Q\} \cdot \mathrm{lcm}\{per(p) \mid p \in Q\}$. We show that $\mathcal{O}(n^7)$ time suffices to compute $\mathrm{lcm}\{per(p) \mid p \in Q\}$ and another number R such that

$$R \leqslant \max\{pre(p) \mid p \in Q\} \leqslant R + |Q|.$$

Consequently, we can also compute a number Z such that $Z \leqslant z \leqslant Z \cdot (|Q|+1)$ in $\mathcal{O}(n^7)$ time (see Theorem 35).

We start with an auxiliary technical lemma.

Lemma 29. *Let $p \in Q$. Let α, β be finite sequences of subsets of Q such that $\mathscr{C}_p = \alpha\beta^\omega$. Then $pre(p) \leqslant length(\alpha)$ and $per(p)$ divides $length(\beta)$.*

Proof. Let α_p, β_p be the unique sequences such that $pre(p) = length(\alpha_p)$, $per(p) = length(\beta_p)$, and $\mathscr{C}_p = \alpha_p \beta_p^\omega$. The fact that $pre(p) \leqslant length(\alpha)$ follows immediately from the definition of α_p. We show that $length(\beta_p)$ divides $length(\beta)$. Let

$$c = (length(\alpha) - length(\alpha_p)) \bmod length(\beta_p)$$

and let γ be the sequence obtained from β_p by 'shifting' the first c elements from its front to its end (for example, if $\beta_p = ABCDE$ and $c = 2$, then $\gamma = CDEAB$). We see that $length(\gamma) = length(\beta_p)$. Moreover, $\mathscr{C}_p = \alpha\gamma^\omega$, which means that $\gamma^\omega = \beta^\omega$. Let S be the first element of γ (and β), and let $i = length(\beta) + 1$. As $\gamma^\omega = \beta^\omega$, the ith element of γ^ω must be S. However, since all elements of γ are pairwise different (this is because the elements of β_p are pairwise different), the ith element of γ^ω can be S only if $i = k \cdot length(\gamma) + 1$ for some $k \in \mathbb{N}_0$. This means that $i = length(\beta) + 1 = k \cdot length(\gamma) + 1$, hence $length(\gamma) = per(p)$ divides $length(\beta)$. □

In the next definition, some of the control states of $\mathscr{A}$ are declared as *repeating*. As we shall see, the prefixes and periods of all repeating states can be computed efficiently; having found their values, it is also possible to determine (or at least estimate) the values of the aforementioned factors which constitute the constant z.

Definition 30. A control state $p \in Q$ is *repeating* if $p \hookrightarrow_j p$ for some $j \in \mathbb{N}$.

Lemma 31. *Let $p \in Q$ be a repeating state. Then $pre(p) < |Q|^2$ and $per(p) < |Q|$.*

Proof. First we show that if p is repeating, then $p \hookrightarrow_i p$ for some $1 \leqslant i \leqslant |Q|$. Let $i \in \mathbb{N}$ be the smallest number such that $p \hookrightarrow_i p$. Due to Lemma 16 we know that there is a chain

$$p = q_0 \hookrightarrow_1 q_1 \hookrightarrow_1 q_2 \hookrightarrow_1 \cdots \hookrightarrow_1 q_i = p.$$

Let us suppose that $i > |Q|$. Then there are $1 \leqslant j < k < i$ such that $q_j = q_k$. This means that there is a shorter chain

$$p = q_0 \hookrightarrow_1 \cdots \hookrightarrow_1 q_j \hookrightarrow_1 q_{k+1} \hookrightarrow_1 \cdots \hookrightarrow_1 q_i = p.$$

From this we obtain $p \hookrightarrow_{i-(k-j)} p$, which contradicts the minimality of i.

Now realize that if there are (some) $j, k \in \mathbb{N}_0$ such that $\mathscr{C}_p(j) \subseteq \mathscr{C}_p(k)$, then also $\mathscr{C}_p(j + l) \subseteq \mathscr{C}_p(k + l)$ for any $l \in \mathbb{N}_0$. This follows easily from the definition of $\mathscr{C}_p$. For the same reason we have that if $\mathscr{C}_p(j) = \mathscr{C}_p(k)$, then $\mathscr{C}_p(j + l) = \mathscr{C}_p(k + l)$ for any $l \in \mathbb{N}_0$. As $\{p\} = \mathscr{C}_p(0) \subseteq \mathscr{C}_p(i)$, we see (due to the previous observation) that

$$\mathscr{C}_p(0) \subseteq \mathscr{C}_p(i) \subseteq \mathscr{C}_p(2i) \subseteq \mathscr{C}_p(3i) \subseteq \cdots \subseteq \mathscr{C}_p(|Q|i).$$

As the length of this non-decreasing chain is $|Q| + 1$, the last two elements (i.e., $\mathscr{C}_p(|Q|i - i)$ and $\mathscr{C}_p(|Q|i)$) must be equal. Hence, if we define α to be the first $|Q|i - i$ elements of $\mathscr{C}_p$, and β the next i elements of $\mathscr{C}_p$, we obtain that $\mathscr{C}_p = \alpha\beta^\omega$. As $i \leqslant Q$, we have that $length(\alpha) < |Q|^2$ and $length(\beta) < |Q|$. These bounds are also valid for $pre(p)$ and $per(p)$ by Lemma 29. □

Lemma 32. *Let $p \in Q$. Then $pre(p) \leqslant |Q|^2 + |Q|$ and $per(p)$ divides* $\mathrm{lcm}\{per(p) \mid p \in Q \text{ is repeating}\}$.

Proof. The case when p is repeating has been handled by Lemma 31. Now suppose that p is not repeating. We show that for each $q \in \mathscr{C}_p(|Q|^2 + |Q|)$ there is a repeating

state r such that $q \in \mathscr{C}_p(|Q|^2 + |Q| + k \cdot per(r))$ for every $k \in \mathbb{N}_0$. It suffices for our purposes, because then we also have that

$$\mathscr{C}_p(|Q|^2 + |Q|) = \mathscr{C}_p(|Q|^2 + |Q| + \mathrm{lcm}\{per(p) \mid p \in Q \text{ is repeating}\}).$$

If we put α to be the first $|Q|^2 + |Q|$ elements of $\mathscr{C}_p$ and β the next $\mathrm{lcm}\{per(p) \mid p \in Q$ is repeating$\}$ elements of $\mathscr{C}_p$, we see that $\mathscr{C}_p = \alpha\beta^\omega$, and thus we get our result by applying Lemma 29.

Let $q \in \mathscr{C}_p(|Q|^2 + |Q|)$. Then there is a chain

$$p = q_0 \hookrightarrow_1 q_1 \hookrightarrow_1 \cdots \hookrightarrow_1 q_{|Q|^2+|Q|} = q.$$

We define r to be the first repeating state q_j which appears in this chain. Then $q \in \mathscr{C}_r(|Q|^2 + |Q| - j)$. Clearly $j \leqslant |Q|$, which also means that $|Q|^2 + |Q| - j \geqslant |Q|^2$. As $pre(r) < |Q|^2$ (due to Lemma 31), q belongs to some element of β_r. Hence, $q \in \mathscr{C}_r(|Q|^2 + |Q| - j + k \cdot per(r))$ for every $k \in \mathbb{N}_0$. Now it suffices to realize that $\mathscr{C}_r(l) \subseteq \mathscr{C}_p(j + l)$ for every $l \in \mathbb{N}_0$. From this we obtain that $q \in \mathscr{C}_p(|Q|^2 + |Q| + k \cdot per(r))$ for every $k \in \mathbb{N}_0$. □

An immediate consequence of Lemma 32 is

Corollary 33. *It holds that*

$$\mathrm{lcm}\{per(p) \mid p \in Q\} = \mathrm{lcm}\{per(p) \mid p \in Q \text{ is repeating}\}.$$

Hence, $\mathrm{lcm}\{per(p) \mid p \in Q\}$ can be computed efficiently; it suffices to evaluate the periods of all repeating states (which are small numbers) and find their least common multiple. Unfortunately, there is no apparent way how to determine the value of $\max\{pre(p) \mid p \in Q\}$. Due to Lemma 32 we know that it is bounded by $|Q|^2 + |Q|$, but even if we construct the first $|Q|^2 + |Q|$ elements of each $\mathscr{C}_p$, it does not help us to recognize the end of α_p (we would have to wait until some element of $\mathscr{C}_p$ repeats, and it leads to the $\mathcal{O}(n^7 + z \cdot n^3)$ bound of Lemma 26). Nevertheless, we can *approximate* the value of $\max\{pre(p) \mid p \in Q\}$. Let $R = \max\{pre(p) \mid p$ is repeating$\}$. The proof of Lemma 32 would also work if we used R instead of $|Q|^2$. Hence we obtain:

Corollary 34. *Let $p \in Q$. Then $pre(p) \leqslant R + |Q|$.*

Now we can formulate and prove the main theorem of this section.

Theorem 35. *Let us define*

$$Z = R \cdot \mathrm{lcm}\{per(p) \mid p \in Q \text{ is repeating}\}.$$

Then Z is computable in $\mathcal{O}(n^7)$ time. Moreover, $Z \leqslant z \leqslant Z \cdot (|Q| + 1)$.

Proof. The fact that $Z \leqslant z$ follows immediately from Corollary 33 and the definitions of z and Z. As each for each $p \in Q$ we have that $pre(p) \leqslant R + |Q|$ (Corollary 34),

	PDA	BPA	OC-A	OC-N
$\approx F$	**PSPACE**-compl. [35,31]	in **P** [33]	**DP**-hard, in **PSPACE**	**DP**-hard, in **PSPACE**
$\sim F$	**PSPACE**-compl. [35,31]	in **P** [33]	in **P**	in **P**
$=_s F$	**EXPTIME**-compl. [30]	**EXPTIME**-compl. [31]	**DP**-hard [27], in **EXPTIME**	in **P** [30]
EF	**PSPACE**-compl. [48]	**PSPACE**-compl. [48]	**DP**-hard [30], in **PSPACE** [48]	**DP**-hard [30], in **PSPACE** [48]
H.M.	**PSPACE**-compl. [34]	**PSPACE**-compl. [34]	in **P**	in **P**

Fig. 5. A summary of known results.

we obtain

$$z \leqslant (R + |Q|) \cdot \mathrm{lcm}\{per(p) \mid p \in Q \text{ is repeating}\}$$

which means that

$$z \leqslant Z + |Q| \cdot \mathrm{lcm}\{per(p) \mid p \in Q \text{ is repeating}\},$$

hence $z \leqslant Z + |Q| \cdot Z$, and thus $z \leqslant Z \cdot (|Q| + 1)$.

To compute Z, we first need to construct $pre(p)$ and $per(p)$ for each repeating state p. Due to Lemma 31 we know that if p is repeating then $pre(p) < |Q|^2$ and $per(p) < |Q|$. Hence, it suffices to construct the first $|Q|^2 + |Q|$ elements of (each) $\mathscr{C}_p$ to find all repeating states together with their prefixes and periods. Due to Lemma 25 we know it can be done in $\mathcal{O}(n^7 + (|Q|^2 + |Q|) \cdot n^3)$ and hence $\mathcal{O}(n^7)$ time. Now we have to compute R, $\mathrm{lcm}\{per(p) \mid p \in Q \text{ is repeating}\}$, and multiply them; it can be of course done in $\mathcal{O}(n^7)$ time. □

4. Conclusions

Recently, the technique used in the proof of Theorem 7 (which was also applied to other problems for OC processes in [30]) was adopted in [27] to show **DP**-hardness of a certain fragment of Presburger arithmetic which is suitable for encoding various problems related to formal verification of OC processes. The main advantage is that the encoding can be then defined by induction on the structure of a (Presburger) formula, and hence the full proof becomes shorter and easier to understand. Thus, the **co-NP** lower bound for the problem of strong bisimilarity between OC-N processes (Theorem 8), as well as the **co-NP** lower bound for the problem of simulation preorder/equivalence between OC-A and FS processes [30], were improved to **DP**.

Currently known results on the complexity of equivalence-checking between (subclasses of) PDA processes and finite-state processes are summarized in the table of Fig. 5. Here, $\approx F$, $\sim F$, and $=_s F$ denote the problem of weak bisimilarity, strong bisimilarity, and simulation equivalence with a finite-state process, respectively. To make the picture more complete, we also added two rows which present complexity

results for the model-checking problem with the Hennessy–Milner (H.M.) logic [37] and the logic EF [13]. The fact that formulae of the H.M. logic can be model-checked in polynomial time for OC-A and OC-N processes follows immediately from the fact that the (in)validity of a H.M. formula φ in a process $p(i)$ of a OC automaton $\mathscr{A}$ depends only on those states of $\mathscr{T}_{\mathscr{A}}$ which are reachable from $p(i)$ in at most $|\varphi|$ transitions. Obviously, there are $\mathcal{O}(|\varphi| \cdot n)$ such states (where n is the size of $\mathscr{A}$) and therefore it is easy to design a polynomial-time model-checking algorithm.

It is quite interesting to compare the complexity issues for BPA (i.e., stateless PDA) processes and OC processes. In some cases, the absence of a finite control unit is a 'stronger simplification' than the replacement of a general stack with a counter; in other cases, however, the opposite is true.

Acknowledgements

I thank the anonymous referee for his many useful comments.

References

[1] P.A. Abdulla, K. Čerāns, Simulation is decidable for one-counter nets, in: Proc. CONCUR'98, Lecture Notes in Computer Science, Vol. 1466, Springer, Berlin, 1998, pp. 253–268.
[2] P.A. Abdulla, M. Kindahl, Decidability of simulation and bisimulation between lossy channel systems and finite state systems, in: Proc. CONCUR'95, Lecture Notes in Computer Science, Vol. 962, Springer, Berlin, 1995, pp. 333–347.
[3] A.V. Aho, J.E. Hopcroft, J.D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1976.
[4] E. Bach, J. Shallit, Algorithmic Number Theory, Vol. 1, Efficient Algorithms, The MIT Press, Cambridge, MA, 1996.
[5] J.C.M. Baeten, J.A. Bergstra, J.W. Klop, Decidability of bisimulation equivalence for processes generating context-free languages, J. Assoc. Comput. Mach. 40 (1993) 653–682.
[6] J.C.M. Baeten, W.P. Weijland, Process Algebra, in: Cambridge Tracts in Theoretical Computer Science, Vol. 18, Cambridge University Press, Cambridge, 1990.
[7] A. Bouajjani, J. Esparza, A. Finkel, O. Maler, P. Rossmanith, B. Willems, P. Wolper, An efficient automata approach to some problems on context-free grammars, Inform. Process. Lett. 74 (2000) 221–227.
[8] A. Bouajjani, J. Esparza, O. Maler, Reachability analysis of pushdown automata: application to model checking, in: Proc. CONCUR'97, Lecture Notes in Computer Science, Vol. 1243, Springer, Berlin, 1997, pp. 135–150.
[9] O. Burkart, D. Caucal, F. Moller, B. Steffen, Verification on Infinite Structures, Handbook of Process Algebra, Elsevier, Amsterdam, 2001, pp. 545–623.
[10] O. Burkart, D. Caucal, B. Steffen, An elementary decision procedure for arbitrary context-free processes, in: Proc. MFCS'95, Lecture Notes in Computer Science, Vol. 969, Springer, Berlin, 1995, pp. 423–433.
[11] D. Caucal, Graphes canoniques des graphes algébriques, Inform. Théor. Appl. (RAIRO) 24 (4) (1990) 339–352.
[12] S. Christensen, H. Hüttel, C. Stirling, Bisimulation equivalence is decidable for all context-free processes, Inform. and Comput. 121 (1995) 143–148.
[13] E.A. Emerson, Temporal and modal logic, Handbook of Theoretical Computer Science, B, Elsevier, Amsterdam, 1991, pp. 995–1072.

[14] J. Esparza, D. Hansel, P. Rossmanith, S. Schwoon, Efficient algorithms for model checking pushdown systems, in: Proc. CAV 2000, Lecture Notes in Computer Science, Vol. 1855, Springer, Berlin, 2000, pp. 232–247.
[15] J. Esparza, J. Knoop, An automata-theoretic approach to interprocedural data-flow analysis, in: Proc. FoSSaCS'99, Lecture Notes in Computer Science, Vol. 1578, Springer, Berlin, 1999, pp. 14–30.
[16] J. Esparza, A. Kučera, S. Schwoon, Model-checking LTL with regular valuations for pushdown systems, in: Proc. TACS'2001, Lecture Notes in Computer Science, Vol. 2215, Springer, Berlin, 2001, pp. 316–339.
[17] J. Esparza, P. Rossmanith, An automata approach to some problems on context-free grammars, in: Foundations of Computer Science, Potential—Theory—Cognition, Lecture Notes in Computer Science, Vol. 1337, Springer, Berlin, 1997, pp. 143–152.
[18] J.F. Groote, A short proof of the decidability of bisimulation for normed BPA processes, Inform. Process. Lett. 42 (1992) 167–171.
[19] J.F. Groote, H. Hüttel, Undecidable equivalences for basic process algebra, Inform. and Comput. 115 (2) (1994) 353–371.
[20] Y. Hirshfeld, M. Jerrum, F. Moller, A polynomial algorithm for deciding bisimilarity of normed context-free processes, Theoret. Comput. Sci. 158 (1996) 143–159.
[21] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, Reading, MA, 1979.
[22] H. Hüttel, C. Stirling, Actions speak louder than words: proving bisimilarity for context-free processes, J. Logic Comput. 8 (4) (1998) 485–509.
[23] P. Jančar, Decidability of bisimilarity for one-counter processes, Inform. and Comput. 158 (1) (2000) 1–17.
[24] P. Jančar, A. Kučera, Bisimilarity of processes with finite-state systems, ENTCS, Vol. 9, 1997.
[25] P. Jančar, A. Kučera, R. Mayr, Deciding bisimulation-like equivalences with finite-state processes, Theoret. Comput. Sci. 258 (1–2) (2001) 409–433.
[26] P. Jančar, A. Kučera, F. Moller, Simulation and bisimulation over one-counter processes, in: Proc. STACS 2000, Lecture Notes in Computer Science, Vol. 1770, Springer, Berlin, 2000, pp. 334–345.
[27] P. Jančar, A. Kučera, F. Moller, Z. Sawa, Equivalence-checking with one-counter automata: a generic method for proving lower bounds, in: Proc. FoSSaCS 2002, Lecture Notes in Computer Science, Vol. 2303, Springer, Berlin, 2002, pp. 172–186.
[28] P. Jančar, F. Moller, Checking regular properties of Petri nets, in: Proc. CONCUR'95, Lecture Notes in Computer Science, Vol. 962, Springer, Berlin, 1995, pp. 348–362.
[29] P. Jančar, F. Moller, Z. Sawa, Simulation problems for one-counter machines, in: Proc. SOFSEM'99, Lecture Notes in Computer Science, Vol. 1725, Springer, Berlin, 1999, pp. 404–413.
[30] A. Kučera, On simulation-checking with sequential systems, in: Proc. ASIAN 2000, Lecture Notes in Computer Science, Vol. 1961, Springer, Berlin, 2000, pp. 133–148.
[31] A. Kučera, R. Mayr, On the complexity of semantic equivalences for pushdown automata and BPA, in: Proc. MFCS 2002, Lecture Notes in Computer Science, Vol. 2420, Springer, Berlin, 2002, pp. 433–445.
[32] A. Kučera, R. Mayr, Simulation preorder over simple process algebras, Inform. and Comput. 173 (2) (2002) 184–198.
[33] A. Kučera, R. Mayr, Weak bisimilarity between finite-state systems and BPA or normed BPP is decidable in polynomial time, Theoret. Comput. Sci. 270 (1–2) (2002) 677–700.
[34] R. Mayr, Strict lower bounds for model checking BPA, ENTCS, Vol. 18, 1998.
[35] R. Mayr, On the complexity of bisimulation problems for pushdown automata, in: Proc. IFIP TCS'2000, Lecture Notes in Computer Science, Vol. 1872, Springer, Berlin, 2000, pp. 474–488.
[36] R. Mayr, Process rewrite systems, Inform. and Comput. 156 (1) (2000) 264–286.
[37] R. Milner, Communication and Concurrency, Prentice-Hall, Englewood, Cliffs, NJ, 1989.
[38] M. Müller-Olm, Derivation of characteristic formulae, ENTCS, Vol. 18, 1998.
[39] Ch. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1994.
[40] D.M.R. Park, Concurrency and automata on infinite sequences, in: Proc. Fifth GI Conf., Lecture Notes in Computer Science, Vol. 104, Springer, Berlin, 1981, pp. 167–183.
[41] W. Reisig, Petri Nets—An Introduction, Springer, Berlin, 1985.

[42] G. Sénizergues, Decidability of bisimulation equivalence for equational graphs of finite out-degree, in: Proc. FOCS'98, IEEE, New York, 1998, pp. 120–129.
[43] J. Srba, Strong bisimilarity and regularity of basic process algebra is PSPACE-hard, in: Proc. ICALP 2002, Lecture Notes in Computer Science, Vol. 2380, Springer, Berlin, 2002, pp. 716–727.
[44] J. Srba, Undecidability of weak bisimilarity for pushdown processes, in: Proc. CONCUR 2002, Lecture Notes in Computer Science, Vol. 2421, Springer, Berlin, 2002, pp. 579–593.
[45] B. Steffen, A. Ingólfsdóttir, Characteristic formulae for processes with divergence, Inform. and Comput. 110 (1) (1994) 149–163.
[46] C. Stirling, Decidability of bisimulation equivalence for normed pushdown processes, Theoret. Comput. Sci. 195 (1998) 113–131.
[47] R. van Glabbeek, The linear time—branching time spectrum, Handbook of Process Algebra, Elsevier, Amsterdam, 1999, pp. 3–99.
[48] I. Walukiewicz, Model checking CTL properties of pushdown systems, in: Proc. FST& TCS 2000, Lecture Notes in Computer Science, Vol. 1974, Springer, Berlin, 2000, pp. 127–138.